REMARKS

Reconsideration of this application is respectfully requested.

As requested, suitable headings have been inserted throughout the specification.

With respect to claim 10, it is noted above that claim 10 was already cancelled in applicant's second preliminary amendment dated June 7, 2000.

The rejection of claims 1-3 under 35 U.S.C. §103 as allegedly being made "obvious" based on Pyne '907 in view of Carson '805 is respectfully traversed.

As will be explained in more detail below, the Examiner appears to have overlooked the fact that applicant's claimed invention requires distribution of workload to the numerous plurality of receiving computers and is otherwise also much more efficient than the approaches taught by either of the cited references (or any conceivable combination of them).

The claimed invention relates to a file synchronization process where a source file at a sending computer is arranged into a sequence of data blocks, each block comprising a predetermined number of data units. A source key value is computed for each block in the source file, and these values are transmitted from the sending computer to the receiving computer. At the receiving computer, reference key values are computed for each predetermined number of contiguous data units in the reference file, and these are

compared with the source key values to determine matches between source key values and reference key values. An indication of which source keys do not have matching reference keys is communicated from the receiving computer to the sending computer. In response, the sending computer transmits data blocks corresponding to the unmatched source keys from the source file to the receiving computer. The receiving computer then constructs a target file that is synchronized with the source file at the sending computer from the contiguous data units in the reference file determined to have reference key values matching respective source key values and the data blocks from the source file received from the sending computer.

Pyne '907 discloses a file synchronization method whereby a reference file on a receiving computer is divided into n-byte data blocks, and a reference key value is associated with each block. The reference key values are transmitted from the receiving computer to the sending computer. At the sending computer, an n-byte block of data in the source file is identified and a source key value is computed for the block. The source key value is compared to the reference key values. If a match is found, an indication of the match is sent from the sending computer to the receiving computer and is used to copy the corresponding block from the reference file to a destination file. If a match is not found, the first byte of the non-matching source file block is sent from the sending computer to the receiving computer, a new n-byte block of data in the source file is identified offset by 1-byte relative to the non-matching block, and a new source key value

is computed for the new block. The source key value is compared to the reference key values as described above, and these steps are repeated until the end of the source file is reached. The destination file thus created is synchronized with the source file.

Carson '805 discloses a file synchronization method whereby a reference file on a destination computer is divided into reference data blocks having a preselected length. Reference key values representing reference blocks of data from the reference file are generated by the destination computer and sent to the source computer. An offset location at the source computer is initialized to point to the start of the source file. The source computer compares a portion of each reference key, referred to as a feature, with portions of the source file. When a feature match is found, a checksum of the reference block (also included in the reference key) is compared with a checksum of the source block. If the checksums also match, then the blocks match. If the matching source block was not found at the current offset location of the source file, then the intervening portion of the source file located before the matching block is sent to the destination computer for inclusion in the synchronized file. A message is then sent to the destination computer so that the matching reference block can be copied from the reference file to build the synchronized file. The offset location at the source computer is updated to point to the end of the matching source block. These steps are repeated until the synchronized file on the destination computer is identical to the source file on the source computer.

With respect to claim 1, the Examiner states:

C

> "The method as taught by Pyne invokes synchronizing date [...] comprising the steps of I) arranging the source file at the sending computer into a sequence of data blocks, each block comprising a predetermined number of data units, and computing a source key value for each block in the source file. (See Pyne's col. 2, lines 22-25)"

However, the referenced sections of Pyne actually show a contrary method whereby the reference file is divided into blocks at the receiving computer, not the source file at the sending computer. There is a similar reversal of roles in relation to setps II) and iii).

In any case, the method of claim 1 is not a simple reversal of Pyne in several important ways, including:

1. Pyne discloses transport of a significantly larger amount of data from the receiving computer to the sending computer than required by the method of claim 1. Modern server computers are not optimized for data traffic in this direction, and hence avoiding the need for such traffic is a significant improvement.

2. Pyne requires the sending computer to perform the computationally expensive step of searching for matching data blocks, whereas the method of claim 1 performs this computation at the receiving computer. Thus the reference file, rather than the source file, is searched for matching blocks. In a typical file server scenario, there are many receiving computers but only one sending computer, so off-loading the computational burden from

the sending computer to the receiving computer is a significant improvement.

3. Pyne requires the sending computer to transmit identification of matched blocks, and also to transmit sequences of single bytes of unmatched data, whereas in the method of claim 1, the ending computer only transmits whole blocks of a predetermined size. This improves performance on the sending computer, in which the source file is already arranged into blocks of a predetermined size.

With respect to claim 2, the Examiner states:

> "As per claim 2, the limitations of: 'wherein the source key values for the sequence of source file data blocks are pre-computed and stored for subsequent use' as specified thereof is present in the proposed combination above (See Pyne's col. 4, lines 1-4)".

However, the cited section of Pyne actually discloses storing "computer program controls that... are stored in RAM and executed by the processing units of each computer." This means storing executable program code implementing the method disclosed in Pyne, not source key values. There is no disclosure of pre-computing and storing source key values for subsequent use.

With respect to claim 3, the Examiner states:

> "As per claim 3, Carson teaches the limitations, wherein the sending computer (source system) and receiving computer (destination system) are

> coupled to communicate by way of an intervening computer containing a cache memory (See Carson's col. 2, lines 48-50), and wherein a copy of the source key values are stored in the intervening computer cache memory and provided therefrom to the receiving computer (See Carson's col. 5, lines 64-67 to col. 6, lines 1-6 and lines 15-25)".

However, column 2, lines 48-50 of Carson disclose using a "cache memory that permits high speed comparisons." This cache memory is high-speed RAM memory of the source computer. There is no disclosure of an intervening computer between the source (i.e., sending) computer and the destination (i.e., receiving) computer. In any case, Carson recites using the cache memory for storing "at least a portion of the source file," not source key values.

In contrast to the citations of the prior art, the cache memory identified in claim 3 is part of an intervening computer such as a network file server between the sending computer and the receiving computer. This intervening computer operates to reduce network data transport.

The rejection of claim 4 under 35 U.S.C. §102 as allegedly anticipated by Carson '805 is also respectfully traversed.

With respect to claim 4, the Examiner states:

> "Carson discloses a similar and method and apparatus for constructing a target data file at a first computer from a reference file stored at the first computer and a source file at a remote second computer such that the constructed target file is synchronized with the source file (See Carson's Abstract), typically the method comprising the steps of:..."

The present application, Pyne and Carson all deal with the problem of constructing, at a first computer having a reference file, a synchronized copy of a source file stored on a remote second computer by identifying which parts of the source file and the reference file are identical and only transferring those parts of the source file which differ. However, each discloses a different method for solving this problem.

The Examiner continues:

> "i) requesting and receiving from the remote second computer a source file summary comprising a sequence of source key values being codes derived from data blocks of predetermined length making up the source file (See Carson's col. 2, lines 23-45);"

However, col. 2, lines 23-45 of Carson actually disclose a contrary method whereby the reference file is divided into reference blocks having a preselected length. Key values derived from these reference data blocks are transferred to the second or source computer. This can be clearly distinguished from the method of claim 4, which provides a number of advantages over the methods of Carson and Pyne, as described below.

The Examiner continues:

> "ii) generating a reference key value for each contiguous portion of the reference source key values, to determine matches therebetween (See Carson's col. 2, lines 46-62, col. 3, lines 14-27 and col. 5, lines 17-29);"

Carson in col. 2, line 47, states that the source file is searched for a group of data units that match the Feature. This method suffers the same problem as that of Pyne,

requiring searching on the sending computer (using a high-demand computational resource), and differs form the method of claim 4, which does not require such searching.

Carson's searching method uses a CRC. The CRC does not characterize the entire data block (so will match the wrong data block on frequent occasions), and does not allow efficient searching for a match beginning at every possible position (each possible first data unit), as does Pyne and the method of claim 4.

Consequently Carson offers to limit the searching to "a portion" of the source file, to reduce the effect of this inefficient search. The effect of limiting the search is simply to reduce the chance of finding a matching block, and instead needing to transmit the block. This reduces the efficiency which is sought.

The likelihood of a false CRC match in Carson is reduced by varying the block size to allow the "Feature" to be varied from one that was "too similar" to another. This reduces the effectiveness of transmission from a block-structured storage medium, an advantage which is provided by the method of claim 4.

The Examiner continues:

> "iii) requesting and receiving from the remote second computer those data blocks from the source file for which no match was found between the corresponding source key value and the reference key values (See Carson's col. 2, lines 63-67 to col. 3, lines 1-13 and col. 6, lines 26-43);"

Carson's col. 2, lines 63-67 further confirm the need for searching "at least the remainder of the portion of the source file". Col. 3, lines 1-13 describe how the ordered list of CRCs is compared with the ordered list of candidate blocks to the end of the list of CRC items and the end of the source file. This comparison is an exhaustive and computationally intensive search using a technique that has been deprecated for decades as being too computationally expensive. It is certainly too expensive for use on a fileserver computer, whereas the method of claim 4 provides efficient performance ona fileserver.

The rejection of claims 5 and 6 under 35 U.S.C. §103 as allegedly being made "obvious" based on Carson '805 in view of Mattis '358 is also respectfully traversed.

Mattis '358 discloses a method for caching and delivering an alternate version of a requested object from a proxy web server on the worldwide web. Alternate versions include different translations of a document, and different versions (e.g., resolutions, colors, etc.) of an image for different web browsers.

With respect to claim 5, the Examiner states:

> "As per claim 5, Carson teaches the claimed limitations: wherein the first and second computers are coupled to communicate over a computer network (See Carson's col. 4, lines 57-67 to col. 5, lines 1-16), and wherein the step of requesting and receiving the source tile [sic] summary includes providing the source file summary to the first computer from a copy of the source file generated at the second computer and previously received (See Carson's col. 6, lines 14-26)."

However, Carson's col. 6, lines 14-26 does not relate to source key values at all, but instead describes the exchange of CRC messages relating to memory resource allocation and is specific to details of Carson's method. In any case, as described above, Carson teaches generating and sending key values for the reference file, not the source file.

The Examiner continues:

> "Carson does not explicitly teaches a method for synchronizing files stored in memory of two remotely system including a proxy computer which is closer or more conveniently located to communicate with the first computer than is the second computer, and stored by the proxy computer. However Mattis teaches a method for caching and delivering an alternate version from among a plurality of alternate versions of information objects (See Mattis's Title and Abstract) and including a proxy computer which is closer or more conveniently located to communicate with the first computer than is the second computer, and stored by the proxy computer (See Mattis's Figs. 1 and 2, and col. 2, lines 7-14)."

It is respectfully submitted that Mattis' method for delivering an alternate version is not applicable here. Neither an incomplete collection of blocks from the source file, nor a summary, nor a list of CRC's or of source or reference file keys, can serve the purpose of the entire original source file. In Mattis, the "alternate versions" are for various browser clients, are include translations of a document to another language (col. 5, lines 20-23) or coding scheme, images which are larger or smaller or have more or fewer colors, or otherwise can substitute for an original file (Mattis, col. 1, lines 36-41) or better meet the client's specific request (Mattis, col. 5, line 24). Each such version is a complete data file. Mattis introduces caches to serve "replicas of popular information

objects" (col. 1, lines 59-60), and the bulk of his innovation is regarding the arrangement of such a cache.

However, neither the incomplete collection of blocks from a source file, nor any of the summary types, is an "alternate version" as used by Mattis. Rather, these are used to discover the extent of differences between the source and reference file, and to determine which sections are similar for synchronization purposes.

Mattis' Figure 1 illustrates the prior art use of an intervening computer or proxy server 30 to serve as a cache (reducing load on the sending computer). Mattis' Figure 2 is a block diagram of the proxy server 30 with cache memory 80. However, Mattis discloses the standard use of a proxy server 30 to serve entire data files requested by a client process. Mattis does not teach or disclose any role for the server 30 in providing file summaries or partial files in a file synchronization (or any other) application.

The prior art methods of synchronization files without communicating the entire source file require active involvement by both the sending and the receiving computer. In contrast, the methods of claims 5 and 6 allow a proxy computer to serve the role of the sending computer, and thus to further reduce the load on the sending computer. This load reduction on the sending computer is a major advantage provided by the methods of claims5 and 6.

The Examiner continues:

> "It would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to modify the teachings of synchronizing files between two remote system as taught by Carson by the proxy server interposed between the clients and the server as taught by Mattis because the proxy provides a middleman gateway service, acting as a server to the client and as client to the server. Therefore the client may be able to access replicas from a topologically proximate cache faster than possible from original web server, while at the same time reducing Internet server traffic."

It is submitted that the methods of claims 5 and 6 are not obvious in relation to synchronizing files by delivering differences only. A proxy cache computer cannot be interposed in either of the methods of Pyne or Carson without the proxy computer becoming a full replica of the sending computer. In contrast, the methods of claims 5 and 6 allow a proxy cache computer which is not a full replica of the sending computer, and may never have a complete copy of any of the source files, to improve data throughput and reduce load on the sending computer.

The none-obviousness of claims 5 and 6 is supported by the fact that such caching is not possible with either Pyne's or Carson's methods.

With respect to claim 6, the Examiner states:

> "As per claim 6, the limitations 'wherein the step iii) of requesting and receiving data blocks for which no match was found includes providing those data blocks to the first computer form the proxy computer from a copy of the source file data blocks previously provided from the second computer and stored in a cache memory at the proxy computer' as specified thereof is present in the proposed combination indicated above (See Mattis's col. 2, lines 7-11 and col. 6, lines 11-17)."

Again, it is submitted that "versions" does not imply "subsets" or summaries, because a subset of the original source file cannot serve the purpose of the entire source file.

The rejection of claims 7, 8 and 9 under 35 U.S.C. §103 as allegedly being made "obvious" based on the three-way combination of Pyne '907, Carson '805 and Mattis '358 is also respectfully traversed.

With respect to claim 7, the Examiner states:

> "As per claim 7, Pyne teaches a method and apparatus for remote file transfer applications (See Pyne's Title). The method as taught by Pyne invokes synchronizing data between a receiving computer and a sending computer, wherein the sending computer has a source file and the receiving computer has a reference file and the receiving and sending computers are coupled for communication therebetween by way of a communications link or network (See Pyne Abstract, Fig. 1 and col. 3, lines 41-50), the method comprising the steps of: i) arranging the source file at the sending computer into a sequence of data blocks, each block comprising a predetermined number of data units, and computing a source key value for each block in the source file (See Pyne's col. 2, lines 22-25)."

This does not describe Pyne's method. Pyne divides the reference file, not the source file, into a sequence of blocks. Pyne discloses a different method, which transmits summaries first from the receiving computer to the sending computer, then computes differences and transmits them from the sending computer to the receiving computer.

The Examiner continues:

> "These passages of Pyne are not explicitly about arranging file, however, Pyne teaches a dividing file in purpose of arranging the file into a plurality of data

blocks; ii) transmitting the source key values form the sending computer to the receiving computer (See Pyne's Abstract and col. 2, lines 47-54; iii) at the receiving computer, comparing the source key values with reference key values computed for each predetermined number of contiguous data units in the reference file to determine matches between source key values and reference key values (See Pyne's col. 2, lines 25-32 and col. 5, lines 37-48); iv) communicating from the receiving computer to sending computer or proxy computer an indication of which source keys do not have matching reference keys, and transmitting data blocks from the source file corresponding to the unmatched source keys from the sending computer or proxy computer to the receiving computer (See Pyne's col. 2, lines 37-46 and col. 5, lines 49-63)."

As above, this does not describe Pyne's method. Moreover, the addition of a proxy computer in the method of claim 7 is not possible with Pyne's method without the proxy computer duplicating all functions of the sending computer, including the ability to compute differences which is not present in existing proxy servers.

The Examiner continues:

"Pyne does not explicitly teach a method for v) constructing at the receiving computer a target file from the contiguous data units in the reference file determined to have reference key values matching respective source key values and the data blocks from the source file received from the sending computer, wherein the constructed target file at the receiving computer is synchronized with the source file at the sending computer. However, Carson, in the same endeavor, teaches a method and apparatus for synchronizing file stored in memory of two remotely located systems (See Carson's Title and Abstract) includes the limitations of: v) constructing at the receiving computer a target file from the contiguous data units in the reference file determined to have reference key values matching respective source key values and the data blocks from the source file received from the sending computer, wherein the constructed target tile at the receiving computer is synchronized with the source tile at the sending computer (See Carson's Fig. 1, col. 3, lines 57-67 to col. 4, lines 1-56 and Appendix A)."

As with Pyne, Carsons' method does not allow a proxy computer to be inserted between the sending and receiving computer without the proxy computer having all functions of a sending computer.

The Examiner continues:

> "The combination of Pyne and Carson does not teach a proxy computer where the sending computer and the proxy computer are couple [sic] for communication therebetween by way of said network. However, Mattis teaches a method for caching and delivering an alternate version from among a plurality of alternate versions of information objects (See Mattis's Title and Abstract) including proxy computer with a cache located in the proxy server that is logically interposed between the client computer and the server computer (See Mattis's Fig. 1 and col. 1, lines 66-67 to col. 2, lines 1-14)."

As described above, Mattis' method for caching is not compatible with either Pyne's nor Carson's method.

With respect to claim 8, the Examiner states:

> "the limitations 'wherein the step of transmitting the source key values from the sending computer to the receiving computer includes storing a copy of the source key values in a cache at the proxy computer and transmitting the source key values from the proxy computer to the receiving computer on request' as specified thereof is present in the proposed combination indicated above (See Mattis's col. 2, lines 6-14).

However, Mattis caching method is not technically combinable with either of Pyne's or Carson's methods. In any case, the method of claim 8 includes storing source key values, not complete files.

With respect to claim 9, the Examiner states:

> "the limitations 'wherein a copy of at least some of the source file data blocks are transmitted from the sending computer and stored in a cache at the proxy computer and wherein the step of transmitting data blocks from the source file includes transmitting said data blocks from the proxy computer cache copy to the receiving computer' as specified thereof is present in the proposed combination indicated above (See Mattis's col. 5, lines 43-67 to col. 6, lines 1-17)."

As described above, Mattis' caching method is not technically combinable with either of Pyne's or Carson's methods. In any case, the method of claim 9 includes storing some of the source file data blocks, not complete files.

The Examiner's attention is also drawn to new claims 11-16. It will be noted that each of the independent claims 11 and 14 clearly distinguishes from any possible teaching or suggestion of any of the cited references (even if combined in any possible way).

Accordingly, this entire application is now believed to be in allowable condition and a formal Notice to that effect is respectfully solicited.

Attached hereto is a marked-up version of the changes made to the claims by the current amendment. The attached page(s) is captioned "**Version With Markings To Show Changes Made**."

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: _____ 25,327
for Larry S. Nixon
Reg. No. 25,640

LSN:vc
1100 North Glebe Road, 8th Floor
Arlington, VA 22201-4714
Telephone: (703) 816-4000
Facsimile: (703) 816-4100